学习笔记（迄 2022.7.25）

一、波士顿房价预测案例相关

波士顿房价预测是较为经典的机器学习案例，其数据内容也内置在了 sklearn 中，该数据集以及其他内置数据集如图一所示，此处可以发现 boston 房价数据集在将来的版本中将会被删除。图一中还有一些不是导入内置数据集的函数。

datasets.load_boston(*[, return_X_y])	已弃用： load_boston 在 1.0 中已弃用，并将在 1.2 中删除。
datasets.load_breast_cancer(*[, return_X_y, ...])	加载并返回乳腺癌威斯康星数据集（分类）。
datasets.load_diabetes(*[, return_X_y, ...])	加载并返回糖尿病数据集（回归）。
datasets.load_digits(*[, n_class, ...])	加载并返回数字数据集（分类）。
datasets.load_files(container_path, *[, ...])	加载具有类别作为子文件夹名称的文本文件。
datasets.load_iris(*[, return_X_y, as_frame])	加载并返回 iris 数据集（分类）。
datasets.load_linnerud(*[, return_X_y, as_frame])	加载并返回体育锻炼 Linnerud 数据集。
datasets.load_sample_image（图像名称）	加载单个样本图像的 numpy 数组。
datasets.load_sample_images()	加载示例图像以进行图像处理。
datasets.load_svmlight_file（F， *[， ...]）	将 svmlight / libsvm 格式的数据集加载到稀疏 CSR 矩阵中
datasets.load_svmlight_files（文件，*[，...]）	从 SVMlight 格式的多个文件中加载数据集
datasets.load_wine(*[, return_X_y, as_frame])	加载并返回 wine 数据集（分类）。

图一

对于将要进行机器学习的数据应进行数据清洗和预处理，其中最重要的是对缺失值的处理。在使用

```
from sklearn.datasets import load_boston
housing=load_boston
```

导入数据集后，housing 对象为一个字典，其中有两个主要的 key 分别为 data、target 分别存放自变量和因变量，将这两部分传递给一个 DataFrame，利用 pandas 模块函数进行数据清晰和预处理。通过其他方法而不是使用 sklearn.datasets 导入的文件同样传递给 DataFrame 后进行处理。

```
import pandas as pd
housingdf=pd.DataFrame(housing.data)
housingdf['y']=(housing.target)
print(housingdf.isna().sum())
```

利用 isna()检验所有位置元素是否缺失，在使用 sum()进行统计。sklearn 内置及常见的波士顿房价数据集，在此处可发现是没有确实值的，对于存在缺失值的数据，简单的处理方法有去除缺失值所在行的数据、利用平均值、中位数填充等。除缺失值，常见预处理对象还有离群值，重复数据等，离群值使用标准差判断，重复数据可使用 pandas 模块 duplicated 函数查找：

```
import numpy as np
def outlier(i):                                          #检查离群值（结果：y 中没有）
    mean=np.mean(i)
    std=np.std(i)
    zscore=(i-mean)/std
    outliers=i[(zscore<-3)|(zscore>3)]
    print(outliers)
outlier(housingdf['y'])
print(housingdf[housingdf.duplicated(keep=False)])       #检查重复值（结果：无）
```

可见波士顿房价数据集是较干净的数据集。

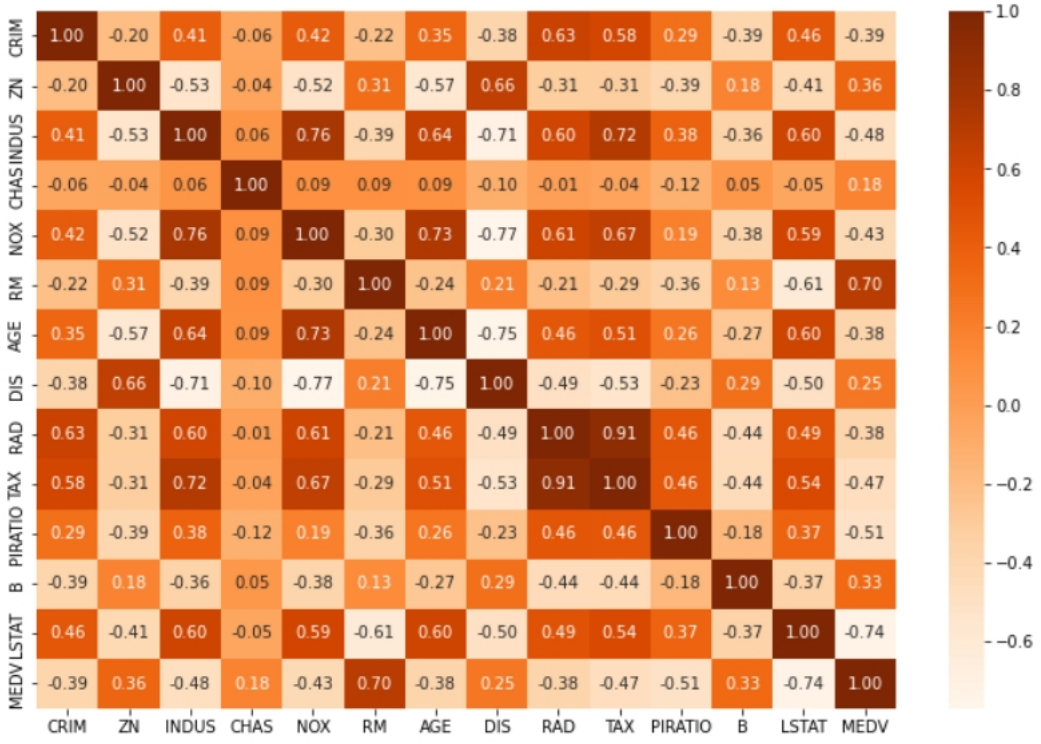
接下来建立回归模型，我使用的 sklearn 版本中波士顿房价数据集（以下简称为“该数据集”）的 data 下共有 13 列，代表 13 种自变量，target 中存有代表房价的因变量。

回归分析是对具有相关关系的两个或两个以上变量之间数量变化的一般关系进行测定，通过一定的数学表达式描述其之间的关系，进而确定一个或几个变量的变化对另一个特定变量的影响程度的统计分析方法。

在进行回归分析之前，应该进行相关性的分析，该数据集中有 506 行数据，利用 t 分布进行相关性的假设检验，或 F 分布进行模型线性关系检验查表时，自由度可以取无限大。

利用 pandas 的 corr()函数，计算每两个变量之间的相关系数，使用 seaborn 绘制热图，从中可以看出因变量与各个自变量之间的相关系数，以及自变量两两之间的相关系数，如图二。

```
Import seaborn
fig=plt.figure(figsize=(12,8))
sns.heatmap(data.corr(), annot=True, fmt='.2f', cmap='Oranges')
sns.pairplot(data,kind='scatter',diag_kind='hist') #kind 类型  diag_kind 对角线类型
```



图二

各变量之间不存在相关性过小或过大的现象，所以我们可以用线性回归模型去预测。

对于标准化处理，使用标准化处理函数 sklearn.preprocessing.StandardScaler() 。

完成数据预处理和对回归分析可行性的验证，开始进行机器学习，使用通用的的机器学习方法步骤：

- 使用 model_selection 中 trian_test_split 函数划分训练集和测试集；
- 导入从对应模块中导入所需模型；
- 交叉验证 cross_val_score 预测、打分；
- 使用 matplotlib,seaborn 等可视化
- 对不同模型得分排序，选择合适模型

我在这里导入了若干机器学习模块：

```
from sklearn.svm import SVR
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import LassoLars
from sklearn.linear_model import BayesianRidge
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
```

从第十四个模块开始，由于其本身一些特性（包括不限于主要用于分类等原因），y 值的训练值只能为整数，使用 astype()函数处理：

```
def work(i,model):
    print(i+1)
    if i<13:
        model.fit(X_train,y_train)
        print(model,' : ', cross_val_score(model,X_test,y_test,cv=5,scoring='r2'))
        dict[list2[i]]=cross_val_score(model,X_test,y_test,cv=5,scoring='r2')
    else:
        model.fit(X_train,y_train.astype('int'))
        print(model,' : ', cross_val_score(model,X_test,y_test.astype('int'),cv=5,scoring='r2'))
        dict[list2[i]]=cross_val_score(model,X_test,y_test,cv=5,scoring='r2')
```

另外该函数使用字典记录了各模型得分，定义各个模型后，分别传入该函数，进行模拟评分：

```
model_1=SVR()
model_2=AdaBoostRegressor()
model_3=RandomForestRegressor()
model_4=ExtraTreesRegressor()
model_5=GradientBoostingRegressor()
model_6=Ridge()
model_7=Lasso()
model_8=ElasticNet()
model_9=LassoLars()
model_10=BayesianRidge()
model_11=LinearRegression()
model_12=DecisionTreeRegressor()
model_13=MLPRegressor()
```

```
model_14=LinearDiscriminantAnalysis()
model_15=LogisticRegression()
model_16=GaussianNB()

list3=[model_1, #0
model_2,#1
model_3,#2
model_4,#3
model_5,#
model_6,
model_7,#5
model_8,#6
model_9,#7
model_10,#8
model_11,#9
model_12,#11
model_13,#12
model_14,#13
model_15,#14
model_16]#15

for i,j in enumerate(list3):
    work(i,j)
    print('\n\n')
```

对得到的字典按 value 值进行排序，查看得分最高的模型：

```
sorteddict=list(dict.items())
sorteddict.sort(key=lambda e:e[1],reverse=True)
```

得到结果如下：

```
[('ExtraTreesRegressor', 0.9362016829598131),
 ('RandomForestRegressor', 0.9290951229757867),
 ('GradientBoostingRegressor', 0.9257777939100139),
 ('MLPRegressor', 0.9097884278070316),
 ('AdaBoostRegressor', 0.893755468383135),
 ('SVR', 0.8753469277000117),
 ('DecisionTreeRegressor', 0.8093068630955981),
 ('LogisticRegression', 0.7952755905511811),
 ('BayesianRidge', 0.7563466928930729),
 ('LinearDiscriminantAnalysis', 0.7559055118110236),
 ('Ridge', 0.7551502652809998),
 ('LinearRegression', 0.7546973359055983),
 ('GaussianNB', 0.5433070866141733),
 ('ElasticNet', 0.26021005187049406),
 ('Lasso', -0.000686336702292456),
 ('LassoLars', -0.000686336702292456)]
```

此情况下（部分模型的默认参数情况）评分由高至低分别是：（0.9 以上：）极端树回归、随机森林回归、梯度提升决策树、多层感知机回归；（0.9 以下，高于 0.3：）AdaBoostRegressor, SVR, DecisionTreeRegressor, LogisticRegression , BayesianRidge ,LinearDiscriminantAnalysis , Ridge , LinearRegression , GaussianNB 。

利用 matplotlib 绘制预测值和真实值散点图，并连接真实值最大值点和最小值点得到参考线段，直观体现预测值偏差。

二、其它一些说明

1. 后缀名为.data 和.csv 的文件，可使用 read_csv 读取
2. 模型的保存：sklearn.externals.joblib

   from sklearn.externals import joblib

   joblib.dump(model,'model.pkl')
3. 模型的读取

   model=joblib.load('model.pkl')

   Y_predict=model.predict()
4. sklearn 三种回归模型的评分：(方法一传入 y 真实值和 y 预测值，其余为 x、y 真实值)

   (1) .metrics 函数 r2_score(y_test,model.predict(X_test))

   (2) .linear_model.LinearRegression 方法 score(X_test,y_test)

   (3) .model_selection 交叉验证函数：

   cross_val_score(model,X_test,y_test,cv=5,scoring='r2') #三者均为 r2 方法
5. 回归系数(多元回归中是一个数组) model.coef_
6. 截距 model.intercept_
7. 查看模型参数 model.get_params()
8. 对标准化的说明：标准化的方法有很多，并且标准化未必使结果得分增加，例如弹性网回归方法和 lasso 回归，在使用未标准化的数据训练的模型得分在 0.7 左右，如图三，而使用经过 sklearn.preprocessing.StandardScaler() 函数标准化的数据得分只有不足 0.3，甚至接近 0。
9. LassoLars 最小角回归在此情况或不适用。
10. Lasso：Least absolute shrinkage and selection operator。

```
housing = load_boston()
X=housing.data
y=housing.target
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
enet = ElasticNet()
model1=Lasso()
model2=LassoLars()
model1.fit(X_train,y_train)
model2.fit(X_train,y_train)
enet.fit(X_train,y_train)
y_predict_enet = enet.predict(X_test)
print('弹性网SCORE:{:.4f}'.format(enet.score(X_test, y_test)))#模型评分
print('Lasso SCORE:{:.4f}'.format(model1.score(X_test, y_test)))
print('LassoLars SCORE:{:.4f}'.format(model2.score(X_test, y_test)))
Run:  未标准化得分高
  warnings.warn(
弹性网SCORE:0.7176
Lasso SCORE:0.7126
LassoLars SCORE:-0.0002
```

图三